Minor in AI

Semi - Supervised Learning

Graph Based Learning

April 09, 2025

1 Helping the New Student Fit In

Sam, a shy student, has just transferred to a new school. On his first day, he walks into a classroom buzzing with chatter and laughter. The teacher gives him a warm welcome, but when Sam looks around, he doesn't really know who he can talk to or where he belongs.

Figure 1: Sam in his New Classroom

Now imagine this classroom as a **graph**.

Each student is a **node**. Some students wear badges — red ones say "I love Math," and blue ones say "Math is not my thing." Only a few wear these badges — most students haven't revealed their preferences yet. These are our **labeled data points**, and the others are **unlabeled**.

Sam doesn't have a badge yet either. But something interesting happens...

He sits beside Aisha, who's kind and always solving puzzles. Aisha wears a red badge. She helps Sam with a Sudoku, and they start chatting.

Soon after, Sam is introduced to Rahul — another friend of Aisha's — who also wears a red badge. Rahul invites Sam to join the Math Club during recess. Sam hesitates but goes along.

As days go by, Sam naturally starts gravitating toward these friends. They do math quizzes together, share tips, and even joke about equations during lunch. Sam begins to *act* like he loves math too, even before he consciously realizes it.

Eventually, if someone were to *guess* Sam's math preference based only on who he spends time with, they'd probably say: "Oh, he's definitely into math!" And they'd likely be right.

This is exactly how **label propagation** works in **graph-based semi-supervised learning**.

We only know the labels of a few nodes. But we *do* know the structure of the graph — who is connected to whom. Over time, the known labels **spread** through the graph, influencing the unlabeled nodes based on their connections. Just like Sam got influenced by Aisha and Rahul, an unlabeled data point can "inherit" the label of its neighbors.

In the real world, these graphs could be:

- Social networks, where users influence each other's opinions.
- Citation networks, where papers share topics based on who they cite.
- E-commerce graphs, where users and products connect via purchases and preferences.

So the question becomes: Can we use these connections to make intelligent guesses about the missing labels?

The answer — thanks to graph-based semi-supervised learning — is a resounding YES!

2 Semi-Supervised Learning

Semi-Supervised Learning (SSL) is a machine learning paradigm where you have a small amount of labeled data and a large amount of unlabeled data.

Think !!!

What if you only had 10 labeled images but 1,000 unlabeled ones? Can we still train a good model?

Graph-based SSL techniques help us tackle this problem by modeling the data as a graph, where:

- Nodes = Data points (labeled + unlabeled)
- Edges = Similarities or relationships between data points
- Edge Weights = Degree of similarity (e.g., Euclidean distance, cosine similarity)

2.1 Core Idea

2.1.1 Assumption

Similar nodes (connected with high edge weights) should have similar labels. This is called the **manifold assumption**.

2.1.2 Goal

Propagate labels from labeled nodes to unlabeled ones using the structure of the graph.

Think !!!

Imagine a social network. If most of a person's friends are movie lovers, the person is likely a movie lover too!

2.2 Steps in Graph-Based SSL

1. Build a graph from the dataset.
2. Assign initial labels to the labeled nodes.
3. Use a propagation algorithm to spread labels to the rest.

2.3 Common Graph Construction Techniques

- **k-Nearest Neighbors (k-NN):** Connect each node to its k nearest nodes.
- **ε-Neighborhood Graph:** Connect nodes that are within a distance ε.
- **Fully Connected Graph:** Connect all nodes with weighted edges (often expensive).

3 Popular Algorithms

1. Label Propagation

This method keeps propagating label probabilities from labeled to unlabeled nodes until convergence. The mathematical expression is given by,

$$F^{(t+1)} = \alpha S F^{(t)} + (1 - \alpha) Y$$

Where:

- $F^{(t)}$: label matrix at iteration t
- S: normalized similarity matrix
- Y: initial labels (only known for labeled nodes)
- $\alpha \in (0, 1)$: balancing parameter

2. Label Spreading

A more regularized variant of Label Propagation, it enforces smoothness and may preserve the original labels more rigidly.

3.1 Key Terms To Remember

- **Harmonic Functions:** Used in label propagation; labels of unlabeled nodes are updated based on neighbors.
- **Manifold Regularization:** Assumes data lies on a low-dimensional manifold; smoothness on the graph is enforced.
- **Graph Laplacian:** $L = D - W$, where D is the degree matrix and W is the weight matrix. Used in many formulations.

4 Example

We are given 6 data points: A, B, C, D, E, F

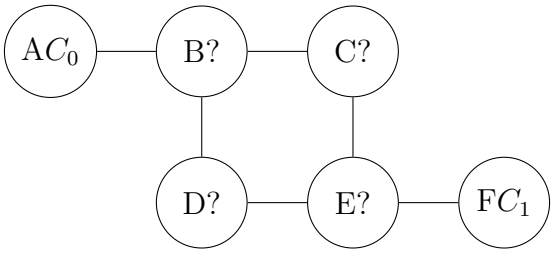
- Points A and F are labeled:

$$\text{Label}(A) = C_0, \quad \text{Label}(F) = C_1$$

- Points B, C, D, E are unlabeled.
- We use 2-Nearest Neighbors to construct a graph.
- Label propagation is used to infer the labels for unlabeled nodes.

4.1 Graph Construction Using 2-NN

Using Euclidean distance (or any similarity metric), we connect each point to its 2 nearest neighbors. Suppose we get the following structure:



This graph connects nodes using their 2-nearest neighbors.

4.2 Step-by-Step Label Propagation

Initial Labels:

$$Y = \begin{cases} C_0 & \text{for node } A \\ C_1 & \text{for node } F \\ ? & \text{for others} \end{cases}$$

Iteration 1:
Each unlabeled node updates its label based on the average label of its neighbors.
Let's assume soft labeling using probabilities:

- Node B is connected to A, C, D → Likely influenced by A (C_0)
- Node C is connected to B, E → B has partial belief in C_0
- Node D is connected to B, E → influenced by both sides
- Node E is between C, D, F → starts feeling pull towards C_1

Iteration 2: Labels get updated again as nodes' neighbors now have more refined label estimates.

Eventually, the propagation converges to something like:

$$\text{Label}(B) = C_0, \quad \text{Label}(C) = C_0, \quad \text{Label}(D) = C_0, \quad \text{Label}(E) = C_1$$

4.3 Key Observations

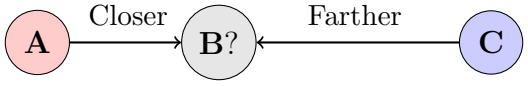
- Labels spread gradually through high-similarity edges.
- Node E switches from ambiguity to C_1 due to proximity to F.
- Node D gets stuck between A and F sides — final label depends on edge weights.

Realization: Label Propagation respects graph structure. Information flows from labeled to unlabeled nodes across paths with strong similarity.

5 DIY Example

Given:

- Points A, B, C in 1D or 2D space
- Labels: $A \rightarrow$ Red, $C \rightarrow$ Blue
- Point B is unlabeled and located **closer to A** than to C



Your Task:

- Using **2-Nearest Neighbors**, build a graph and try **Label Propagation**.
- Predict what label B would likely receive.
- Try coding this in Python with scikit-learn or simulate updates step-by-step.

What color will B get using Label Propagation?
Hint: Since B is closer to A (Red), label propagation will likely assign B a Red label. Distance and edge weight matter. Use Gaussian kernel for weights:

$$w_{ij} = \exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma^2}\right)$$

6 Key Takeaways

1. SSL Graph-based methods are powerful when labels are scarce.
2. They exploit similarity and structure in data to generalize.
3. Understanding Python behavior can help debug graph-like models!